Hardware-Efficient Transformer Training via Piecewise Affine Operations

Atli Kosson & Martin Jaggi
Machine Learning & Optimization Laboratory

Motivation

Problem: Neural network training is computationally expensive with the multiply-and-accumulate operations performed in matrix multiplications accounting for most of this cost. Compared to addition, multiplication is significantly more demanding in terms of hardware area and energy and is ultimately the primary contributor to the overall computational burden.

Questions:

- Can we replace matrix multiplications with theoretically cheaper piecewise-affine alternatives?
- Can neural networks be trained in a completely "multiplication-free" manner by extending this approach to other operations (see figure ➞)?

Background: Floating Point Numbers

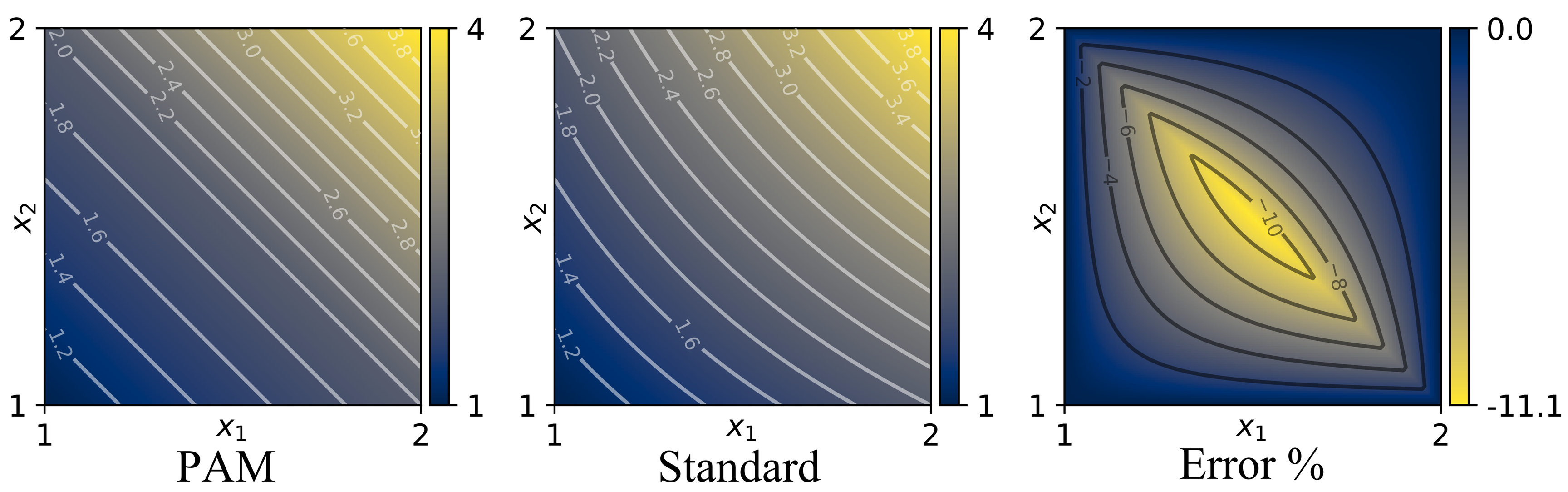
- Binary equivalent of the scientific notation:
  $$(-1)^S \cdot 2^E \cdot (1+M) \text{ with } S \in \{0,1\},\ E \in \mathbb{Z},\ 0 \le M < 1$$
- The bit representation of this using IEEE 754 float32 encoding is:
  $$[S, e_0, \ldots, e_7, m_0, \ldots, m_{22}]$$
  with $\bar{E} = (e_0 \ldots e_7)$, $E = \bar{E} - 127$, $\bar{M} = (m_0 \ldots m_{22})$, $M = \bar{M}/2^{23}$
- Special interpretations for $\bar{E} = 0$ and $\bar{E} = 255$ give 0, NaN and $\pm\infty$

Piecewise-Affine Multiplication

Definition:
Let $A = (-1)^{S_A} \cdot 2^{E_A} \cdot (1+M_A)$ and $B = (-1)^{S_B} \cdot 2^{E_B} \cdot (1+M_B)$
Piecewise-Affine Multiplication (PAM) is then defined as $C = A \hat{\cdot} B$ with:
$$C = (-1)^{S_C} \cdot 2^{E_C} \cdot (1+M_C)$$
$$S_C \coloneqq (S_A + S_B) \bmod 2$$
$$E_C \coloneqq E_A + E_B + 1\{M_A + M_B \ge 1\}$$
$$M_C \coloneqq M_A + M_B - 1\{M_A + M_B \ge 1\}$$
where $1\{a\}$ is one when a holds and zero otherwise.

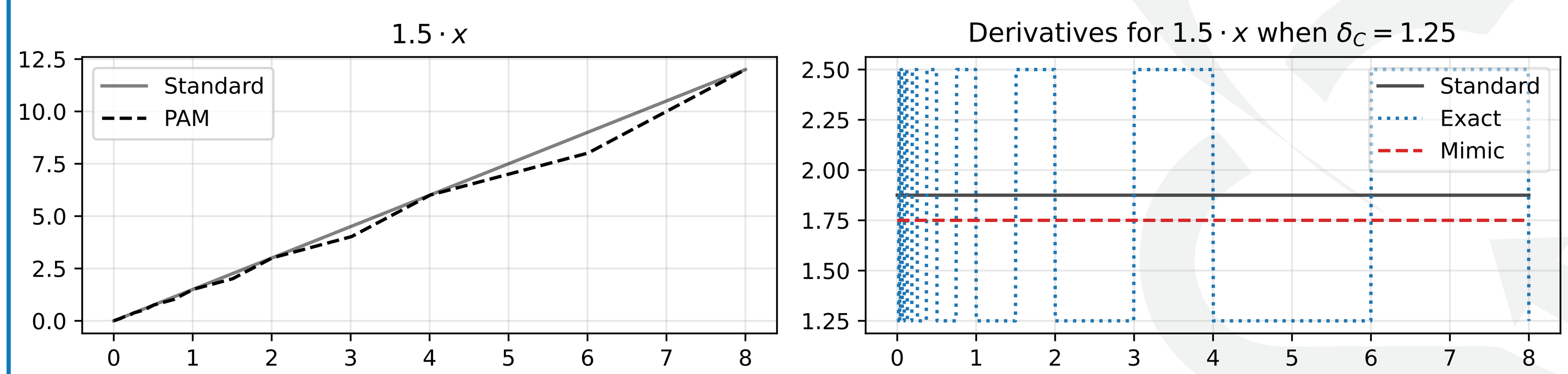
Implementation:
Adding the binary representations together as int32 which allows $\bar{M}$ to overflow into $\bar{E}$. Then correcting for the exponent offset of -127 and handling edge cases including exponent over/underflows and NaN/Inf.

Approximates standard multiplication (pattern repeats per binade):



Two types of derivatives:

- **Exact bwd:** $\delta_A = 2^{E_B + 1\{M_A + M_B \ge 1\}} \delta_C$, the true derivative
- **Mimic bwd:** $\delta_A = B \hat{\cdot} \delta_C$, uses PAM in standard derivative formula
- Both can be performed without standard multiplications.
- The mimic derivative is biased but has less variance.



Other Piecewise-Affine Operations

- We can construct other piecewise-affine operations e.g., log/exp
  $\mathrm{paexp}_2(A) = 2^{\lfloor A \rfloor} \cdot (1 + A - \lfloor A \rfloor)$, $\quad \mathrm{palog}_2(A) = E_A + M_A, A > 0$
- Operations like sqrt, exp etc. can be defined in terms of these
- Also have two types of derivatives, exact and a mimic version based on the formula for the derivative of the standard analog

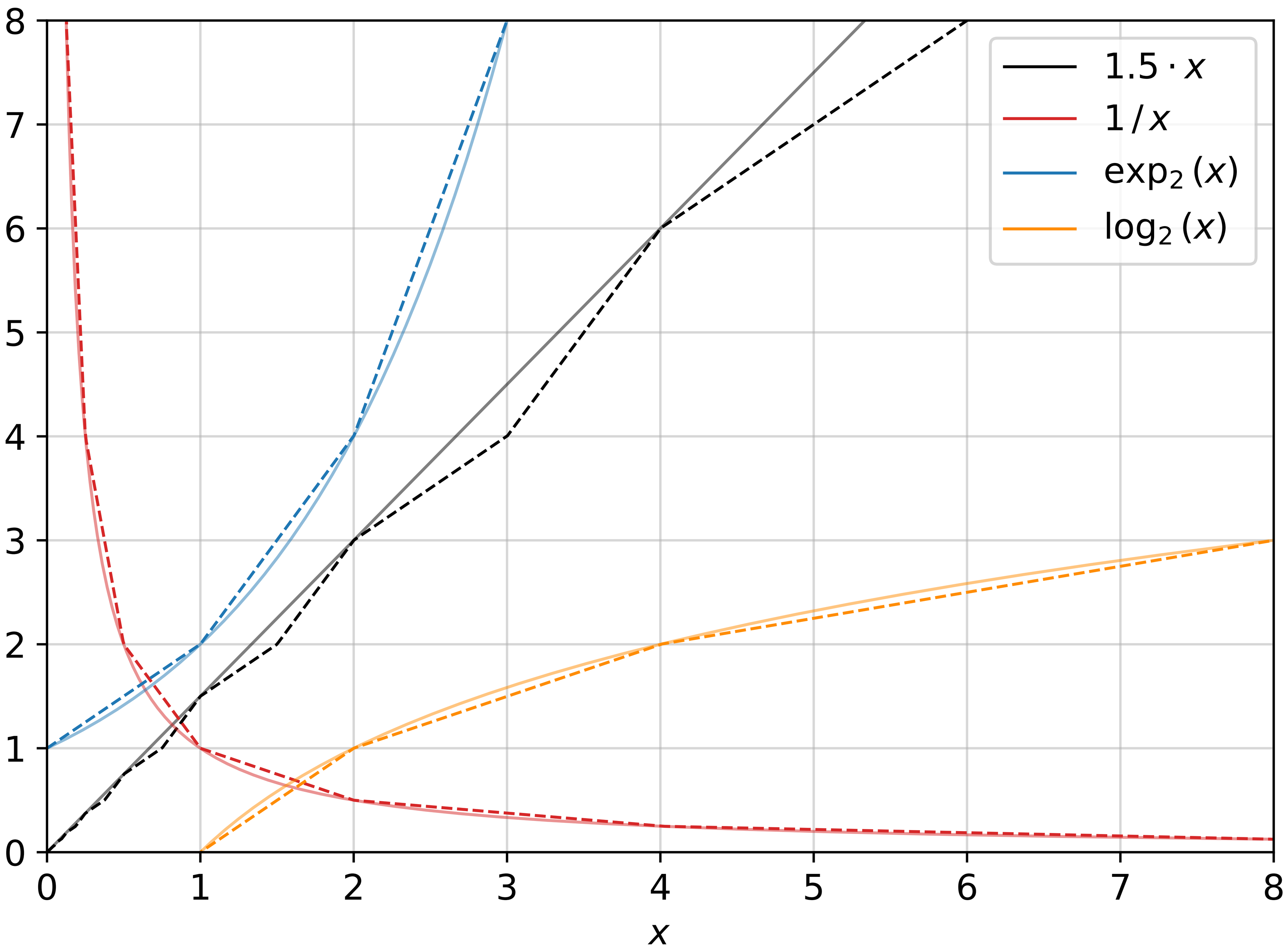


Figure: Elementary neural network operations (solid lines) along with cheaper piecewise affine alternatives (dashed lines). The piecewise affine functions periodically match the originals, while the slope of each segment is an exact power of 2.

Experiments

Piecewise-Affine Matrix Multiplications

	Dataset	Baseline	PA-Matmul	Adder (Alternative method)
DeiT-Tiny	CIFAR10	94.8%	95.0[+0.2]%	92.4[−0.2]%*
	ImageNet	72.2%	72.2[+0.0]%	70.5[−1.7]%

	Network	Baseline	PA-Matmul
ConvNets (CIFAR-10)	VGG-13	$92.9 \pm 0.3\%$	$92.9[+0.0] \pm 0.2\%$
	ResNet-20	$92.1 \pm 0.3\%$	$92.0[-0.1] \pm 0.2\%$
	ResNet-110	$94.2 \pm 0.1\%$	$94.1[-0.1] \pm 0.2\%$
	ResNeXt-20 4x16	$93.7 \pm 0.2\%$	$93.8[+0.1] \pm 0.2\%$
	ConvMixer-256/8	$94.8 \pm 0.2\%$	$94.8[-0.0] \pm 0.1\%$

Narrow Mantissas

Matmul Type	VGG-13 CIFAR-10 Test Accuracy	Transformer-Small IWSLT14 BLEU Score
float32	$92.9 \pm 0.3\%$	34.4 ± 0.1
PAM float32	$92.9 \pm 0.2\%$	34.2 ± 0.2
PAM bfloat	$92.9 \pm 0.2\%$	34.4 ± 0.2
PAM 4 bit mantissa	$92.9 \pm 0.5\%$	34.2 ± 0.1
PAM 3 bit mantissa	$92.8 \pm 0.2\%$	29.4 ± 0.5

Fully Multiplication-Free Training

IWSLT14 DE-EN BLEU Score

PA Operation(s)	Exact bwd	Mimic bwd	Cumulative
Matmul	$33.4[-1.0] \pm 0.1$	$34.2[-0.2] \pm 0.2$	$34.2[-0.2] \pm 0.2$
Attention Softmax	$3.0[-31.4] \pm 0.9$	$34.3[-0.1] \pm 0.7$	$34.2[-0.2] \pm 0.2$
Layer Norm	$33.7[-0.7] \pm 0.2$	$34.4[-0.0] \pm 0.1$	$34.4[-0.0] \pm 0.1$
Loss	$33.7[-0.7] \pm 0.2$	$33.0[-1.4] \pm 0.9$	$33.7[-0.7] \pm 0.2$
Optimizer	$34.2[-0.2] \pm 0.1$		$33.5[-0.9] \pm 0.3$

Conclusion: **Able to match the baselines with piecewise-affine matrix multiplications. Only minor degradation replacing all multiplications.**

PAM Hardware Cost

Format	Addition Energy [pJ]	Addition Area [µm²]	Multiplication Energy [pJ]	Multiplication Area [µm²]
int32	0.1	137	3.1	3495
int16	0.05	67		
int8	0.03	36	0.2	282
float32	0.9	4184	3.7	7700
float16	0.4	1360	1.1	1640

PAM vs standard multiplication for float32:

- ~20x energy savings, ~30x area savings

Special hardware required to realize gains

arXiv:2305.17190